



CENTRO UNIVERSITÁRIO GERALDO DI BIASE
FUNDAÇÃO EDUCACIONAL ROSEMAR PIMENTEL

Apostila de Introdução a Computação

SISTEMAS OPERACIONAIS

Prof. Eng. Wellington Vergilio Fortes

1. Conceituação

O Sistema Operacional, também chamado de “programa básico”, constitui-se de um conjunto de módulos de programas que, dentro da visão em hierarquia de camadas, mostrada na figura abaixo, servem como interface entre os programas aplicativos ou programas utilitários e o hardware do sistema computacional.

Cabe ao sistema operacional:

- a) controlar todos os recursos do sistema computacional hospedeiro: processador, subsistemas de memória, dispositivos periféricos e até mesmo os arquivos armazenados na memória principal ou em memória secundária;
- b) garantir que os diversos usuários de um sistema computacional e os diferentes processos que nele estejam sendo executados não interfiram de forma indesejável entre si (respeitando os respectivos espaços alocados na memória principal e o acesso aos dispositivos de entrada/saída).
- c) Assegurar a segurança do sistema computacional, negando o acesso a usuários não autorizados.

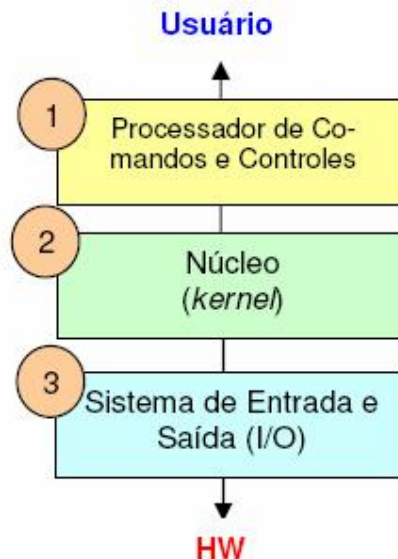


2. Estrutura básica de um sistema operacional

O sistema operacional utilizado em máquinas de uso geral, em sua forma mais simples, pode ser apresentado como composto por 3 blocos, ou camadas:

1) Processador de Comandos e Controles (Shell): recebe os comandos introduzidos pelos usuários, interpreta-os e, uma vez que os considera válidos, providencia sua execução valendo-se das rotinas que constituem os blocos 2 e 3 (mais próximas do nível da máquina). Caso o comando recebido não seja considerado válido, emite uma mensagem de erro para o usuário.

2) Núcleo, ou kernel: é o módulo central do sistema operacional, que é carregado em primeiro lugar na Memória Principal e aí permanece. Normalmente, o núcleo do SO é responsável pelo controle dos processos que estão sendo executados, pelo gerenciamento de memória (alocação de memória), e pelo gerenciamento de espaço em disco (criação, abertura, gravação, leitura, remoção e fechamento de arquivos).

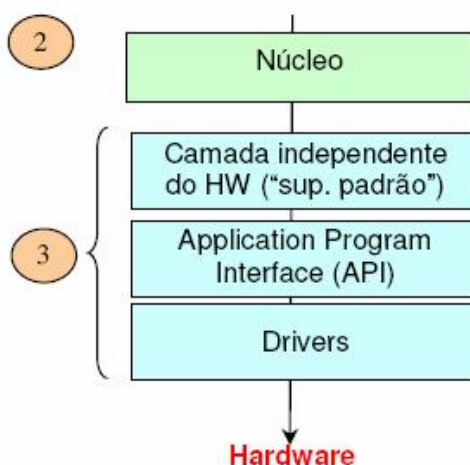


3) Sistema de Entrada/Saída: responsabiliza-se pela execução física das operações de e/s, realizando o acesso aos periféricos. No caso dos discos, cria um “disco lógico” para ser empregado pelos níveis superiores, com seus endereços internos mapeados no disco físico existente. Normalmente esta parte do sistema operacional fica residente em memória ROM, para que não seja afetada por eventual avaria de disco, que impediria a inicialização (ou IPL – “Initial Program Loading”) do sistema computacional.

O Sistema de E/S não precisa ser específico para determinada máquina (considerando-se que é possível utilizar o mesmo Sistema Operacional para mais de um hardware). É criada uma “superfície padrão” no nível hierárquico do Sistema de E/S, que pode ser usada pelos componentes de hierarquia superior do Sistema Operacional e realiza serviços gerais de entrada e saída, tais como atribuição de prioridade de acesso e alocação de periféricos e “spooling”.

A camada de “*Application Program Interface*” conta com rotinas, protocolos e ferramentas para construção de aplicações.

A camada de *drivers* é constituída de rotinas para acesso e controle dos periféricos, com o tratamento que cada um requer (blocos de caracteres ou caracteres individuais). Esses *drivers* traduzem os comandos recebidos do nível superior para os comandos específicos do periférico endereçado. Embora muitos desses *drivers* já venham com o sistema operacional, outros poderão ser fornecidos pelo fabricante do periférico.



3. Chamadas o Sistema Operacional

Os utilizadores do sistema computacional comunicam-se com o Hardware por meio dos Programas do Usuário ou por meio do próprio Processador de Comandos e Controles. Ambos terão a mesma precedência hierárquica e farão chamadas a rotinas existentes nos blocos mais baixos do sistema operacional.

a) Exemplos de Rotinas do núcleo

- Criação de arquivo;
- Extinção de arquivo;
- Abertura de arquivo;
- Fechamento de arquivo;
- Leitura de arquivo;

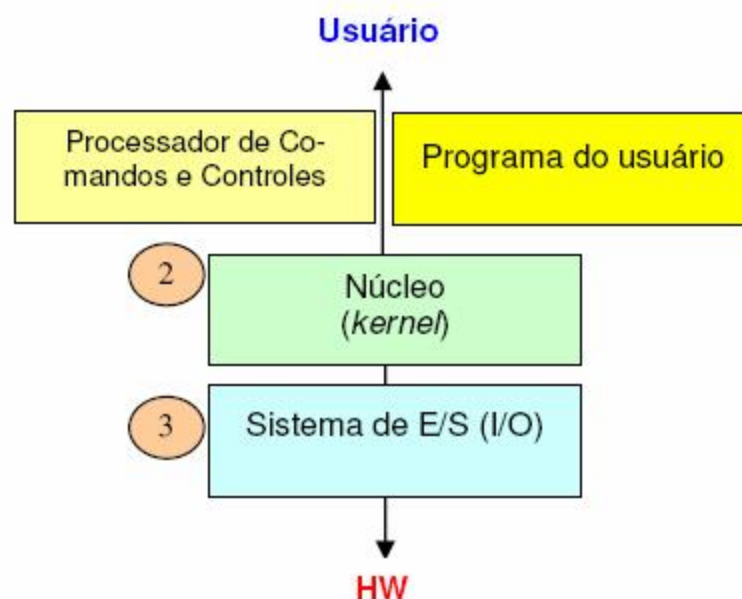
b) Exemplos de rotinas do Sistema de E/S

- Verificar status do teclado;
- Obter caracter do teclado;
- Enviar caracter para a tela do monitor;
- Enviar caracter para a impressora;
- Acesso a disco – vejamos com mais detalhe esta chamada:

Pode-se imaginar que o programa que faz a chamada forneça as seguintes informações:

Número do disco, trilha, setor, endereço de memória (informado pela UCP ao DMA), operação (entrada ou saída). Neste caso, os níveis hierárquicos acima do Sistema de E/S precisam saber das características do dispositivo, pois os setores de discos de diversas procedências não são uniformes.

Assim, o Sistema de E/S procura simular para o Núcleo um disco padrão, com propriedades que não necessariamente serão as do disco físico existente. Ou seja: é criado um “disco lógico”, para emprego pelos níveis superiores, em que seus endereços são mapeados no disco físico existente.

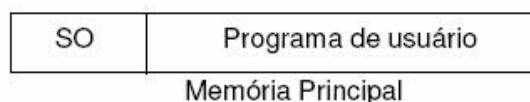


4. Tipos de Sistemas Operacionais

a) Mono-usuário

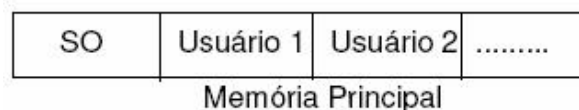
Praticamente não são mais usados em computadores de emprego geral, embora ainda sejam empregados em máquinas especiais, destinadas a controle de processos.

Eram utilizados em sistemas computacionais (normalmente micros) sem proteção de memória, sem relocação dinâmica de memória, sem estados supervisor/usuário, que faziam apenas operações de E/S programada. Apenas um usuário podia utilizar o computador. A área da MP não ocupada pelo sistema operacional era alocada a apenas um processo.



b) Multi-usuário

Usados em sistemas computacionais que permitem que vários usuários executem seus processos (aparentemente) ao mesmo tempo. Em verdade, esses processos concorrem ao controle da UCP, segundo políticas estabelecidas pelo sistema operacional que estabelecem prioridades e intervalos de tempo para cada um.



c) De rede

Um sistema operacional de rede deve possuir suporte à operação em rede, ou seja, a capacidade de oferecer às aplicações locais recursos que estejam localizados em outros computadores da rede, como arquivos e impressoras. Ele também deve disponibilizar seus recursos locais aos demais computadores, de forma controlada. A maioria dos sistemas operacionais atuais oferece esse tipo de funcionalidade.

d) Distribuído

Em um sistema operacional distribuído, os recursos de cada máquina estão disponíveis globalmente, de forma transparente aos usuários. Ao lançar uma aplicação, o usuário interage com sua janela, mas não sabe onde ela está executando ou armazenando seus arquivos: o sistema é quem decide, de forma transparente. Os sistemas operacionais distribuídos já existem há tempos (Amoeba [TKvRB91] e Clouds [DRJLAR91], por exemplo), mas ainda não são uma realidade de mercado.

e) Servidor

Um sistema operacional servidor deve permitir a gestão eficiente de grandes quantidades de recursos (disco, memória, processadores), impondo prioridades e limites sobre o uso dos recursos pelos usuários e seus aplicativos. Normalmente um sistema operacional servidor também tem suporte a rede e multi-usuários.

f) Embutido

Um sistema operacional é dito embutido (*embedded*) quando é construído para operar sobre um hardware com poucos recursos de processamento, armazenamento e energia. Aplicações típicas desse tipo de sistema aparecem em telefones celulares, controladores industriais e automotivos, equipamentos eletrônicos de uso doméstico (leitores de DVD, TVs, celulares, centrais de alarme, etc).

Muitas vezes um sistema operacional embutido se apresenta na forma de uma biblioteca a ser ligada ao programa da aplicação (que é fixa). Exemplos de sistemas operacionais embutidos são o Symbian, Xylinx, LynxOS e VxWorks.

g) Tempo real

Ao contrário da concepção usual, um sistema operacional de tempo real não precisa ser necessariamente ultra-rápido; sua característica essencial é ter um comportamento temporal previsível (ou seja, seu tempo de resposta deve ser conhecido no melhor e pior caso de operação). A estrutura interna de um sistema operacional de tempo real deve ser construída de forma a minimizar esperas e latências imprevisíveis, como tempos de acesso a disco e sincronizações excessivas.

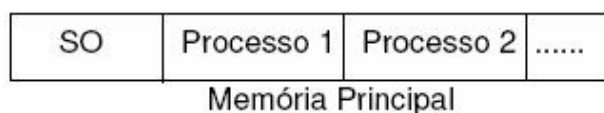
Existem duas classificações de sistemas de tempo real: *soft real-time systems*, nos quais a perda de prazos implica na degradação do serviço prestado. Um exemplo seria o suporte à gravação de CDs ou à reprodução de músicas. Caso o sistema se atrase, pode ocorrer a perda da mídia em gravação ou falhas na música que está sendo tocada. Por outro lado, nos *hard real-time systems* a perda de prazos pelo sistema pode perturbar o objeto controlado, com graves consequências humanas, econômicas ou ambientais.

Exemplos desse tipo de sistema seriam o controle de funcionamento de uma turbina de avião a jato ou de uma caldeira industrial.

Exemplos de sistemas de tempo real incluem QNX, RT-Linux e VxWorks. Muitos sistemas embutidos têm características de tempo real, e vice-versa.

h) Multi-programado (multi-task)

Usados em sistemas computacionais que permitem que vários processos sejam carregados em memória e concorram pelo controle da UCP, segundo políticas estabelecidas pelo sistema operacional, que estabelecem prioridades e intervalos de tempo para cada um. Este tipo de sistema operacional pode ser subdividido em dois subtipos:



1) Cooperativo

Determinado processo, uma vez recebendo o controle da UCP, permanece com ele o tempo que lhe for necessário. Era usualmente empregado em sistemas computacionais de menor capacidade (micros). Um bom exemplo deste tipo de sistema operacional é o MS Windows 3.1, em que podíamos carregar vários programas em memória, mas apenas um deles constituía-se em processo ativo.

b) Preemptivo

Neste subtipo, o sistema operacional estabelece políticas que evitam que um dos processos monopolize o uso da UCP, permitindo que todos possam concorrer a seu controle. Há mais de uma política destinada a permitir que todos os processos tenham acesso ao controle da UCP, como veremos a seguir. Podem ser citados como exemplos de SO multiprogramados preemptivos o UNIX e o Windows 98.

5. Políticas de administração do acesso dos processos à UCP

a) Prioridades:

Os processos, ao serem carregados, recebem uma prioridade para sua execução, que poderá refletir a urgência do trabalho que realizarão, ou o prestígio, dentro da organização, do setor a que o processo esteja afeto.

Pode haver um outro critério na aplicação desta política, mais preocupado com a eficiência do sistema computacional, considerando a natureza dos processos, que serão classificados em 2 tipos, segundo o uso previsível que farão da UCP. Os processos “*i/o bound*” são aqueles que fazem pouco uso da UCP e realizam de forma mais intensiva operações de E/S (mais freqüentemente encontrados em aplicações comerciais). Os processos “*cpu bound*” são aqueles que empregam intensivamente a UCP, realizando poucas operações de E/S (mais comuns em aplicações científicas). Neste critério, atribui-se maior prioridade aos processos “*i/o bound*”, que passam a ser executado sem primeiro lugar (foreground).

b) Equidade:

Em determinados ambientes, não haverá razão para que um processo tenha prioridade sobre outro. Assim, atribui-se a todos os processos “fatias de tempo” (*timeslices*) iguais, que são medidas pelo relógio de tempo real ou por relógio de intervalo do computador.

c) Adaptação automática ao comportamento do programa

Nesta política, todos os processos carregados recebem uma “fatia de tempo” e um indicador de prioridade. Caso o processo solicite uma operação de E/S antes de acabar o tempo que lhe foi destinado, sua prioridade é aumentada e sua “fatia de tempo” é diminuída (processo “*i/o bound*”). Caso o tempo alocado ao processo expire antes que esse solicite operação de E/S, sua prioridade é diminuída e sua “fatia de tempo” aumentada (processo “*cpu bound*”). Depois de algum tempo, os processos que usam mais intensivamente a UCP passam a ser executados em “background”: com mais baixa prioridade e com mais tempo de controle do processador. Já aqueles que utilizam mais E/S, são executados em “foreground”: com prioridade maior, mas com menos tempo de controle da UCP.

6. Estados dos Processos

Um processo carregado em memória, concorrendo ao controle da UCP, poderá estar em um dos três estados abaixo:

CENTRO UNIVERSITÁRIO GERALDO DI BIASE FUNDAÇÃO EDUCACIONAL ROSEMAR PIMENTEL

- *Pronto (ready)* – o processo está em condições de assumir o Controle da UCP. Apenas, em obediência à política de compartilhamento implantada pelo SO.
- *Executando (executing)* – o processo está com o controle da UCP. Em sistemas computacionais dotados de apenas uma UCP, apenas 1 processo, em determinado instante, poderá estar neste estado.
- *Bloqueado (hold)* – o processo não está, temporariamente, concorrendo ao controle da UCP, pois lhe falta algum elemento necessário (por exemplo, o resultado de uma operação de E/S). O processo ficará neste estado até que o dispositivo que fazia a operação esteja pronto para voltar a ficar no est

